Cloud File Keeper

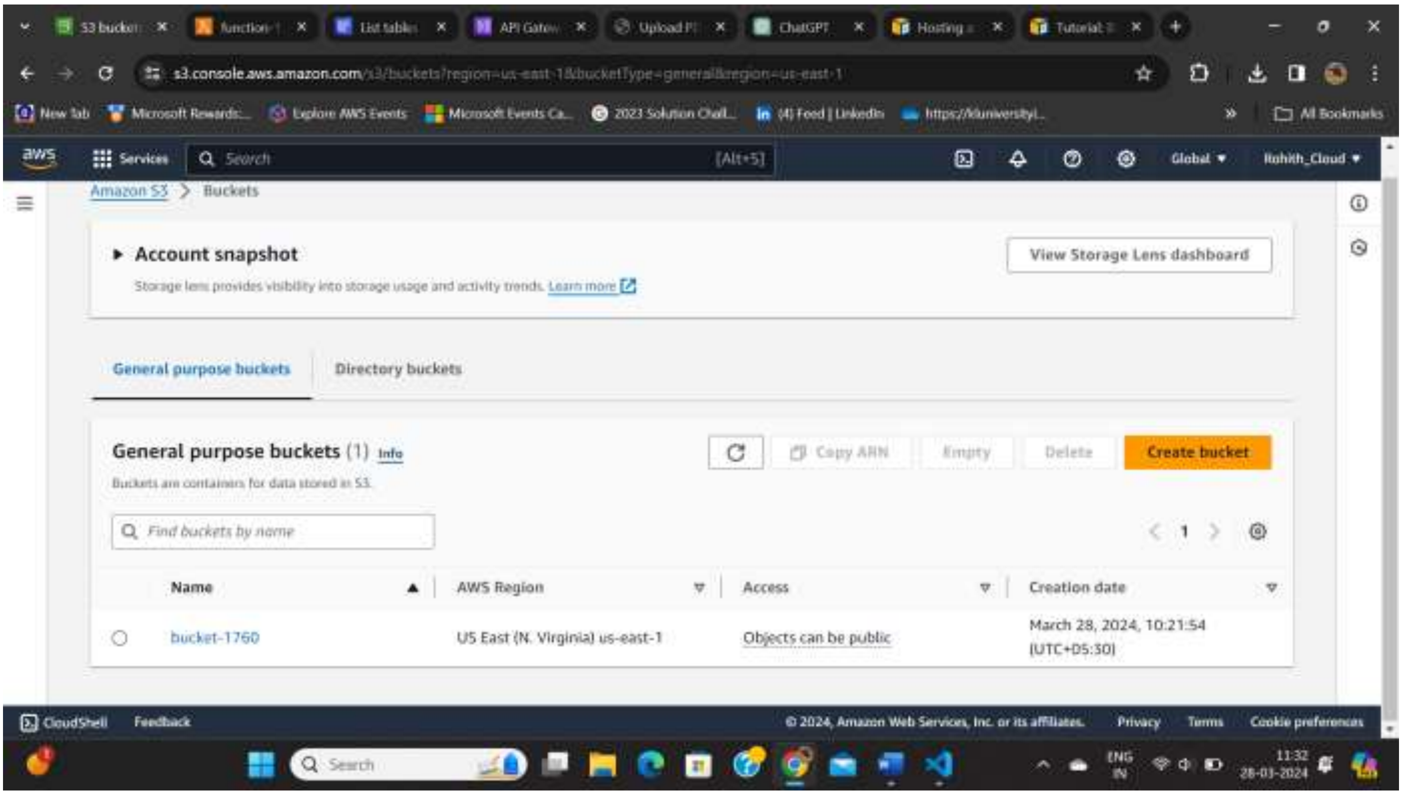
Abstract:

Create a dynamic website for file management using AWS services. Utilize S3 for website hosting, DynamoDB for storing file metadata, API Gateway for Lambda endpoint integration, and Lambda for backend logic. Develop frontend functionality for file upload and implement CRUD operations through API calls. Ensure proper CORS configuration for seamless interaction between frontend and backend components. Test thoroughly before deploying to S3 bucket, and monitor performance post-deployment for seamless file management experience.

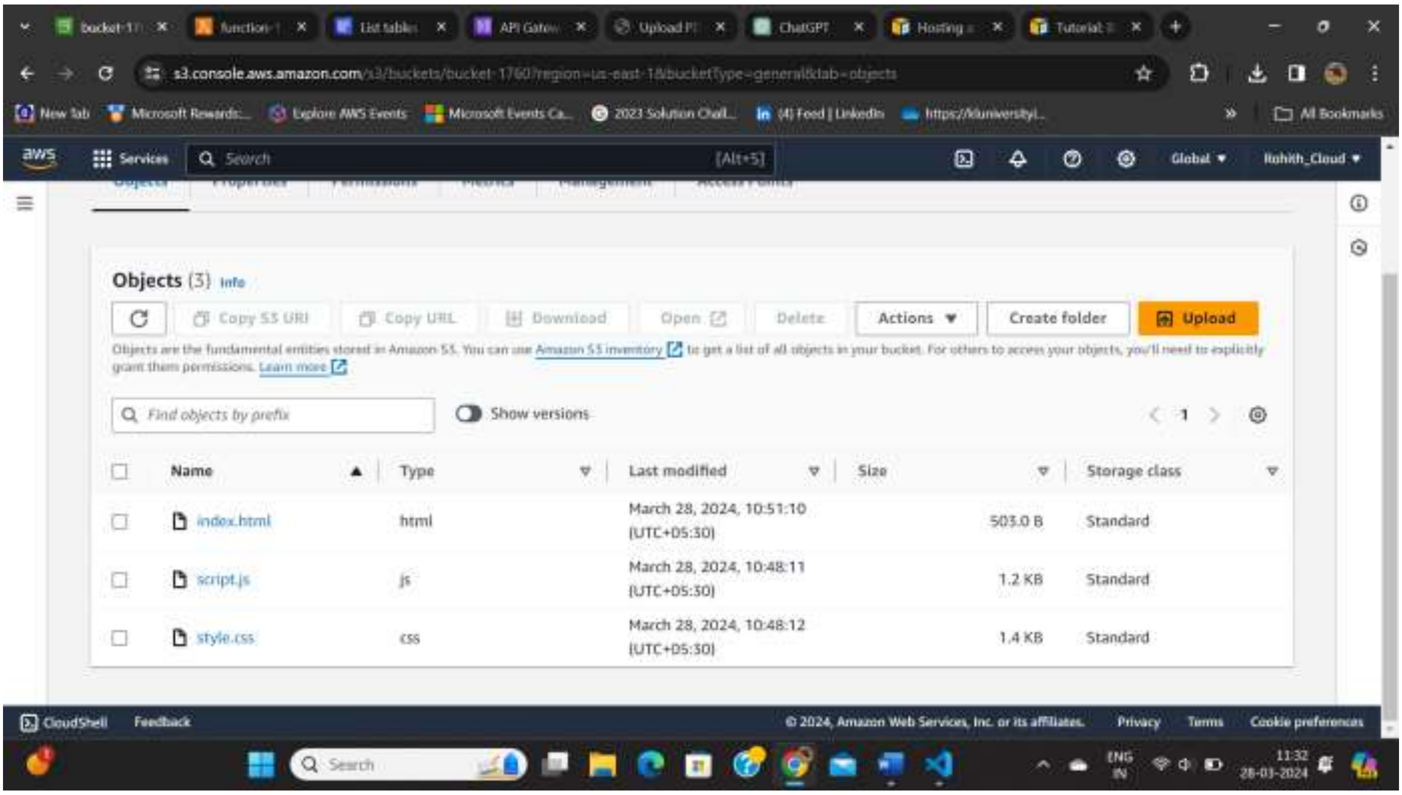
Services Used: S3, DynamoDB, Lambda, API Gateway, IAM Role

Steps:

1. Create a S3 bucket, with the desired location to upload our website content
2. Make the bucket and it's objects public
3. Enable static website hosting for your website


s3.console.aws.amazon.com/s3/buckets?region=us-east-1&bucketType=general&region=us-east-1
aws
Services
Search
[Alt+S]
Global
Rohith_Cloud
Amazon S3 > Buckets
Account snapshot
Storage lens provides visibility into storage usage and activity trends. Learn more
View Storage Lens dashboard
General purpose buckets
Directory buckets
General purpose buckets (1) Info
Buckets are containers for data stored in S3.
Copy ARN
Empty
Delete
Create bucket
Find buckets by name
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

	Name	AWS Region	Access	Creation date
○	bucket-1760	US East (N. Virginia) us-east-1	Objects can be public	March 28, 2024, 10:21:54 (UTC+05:30)

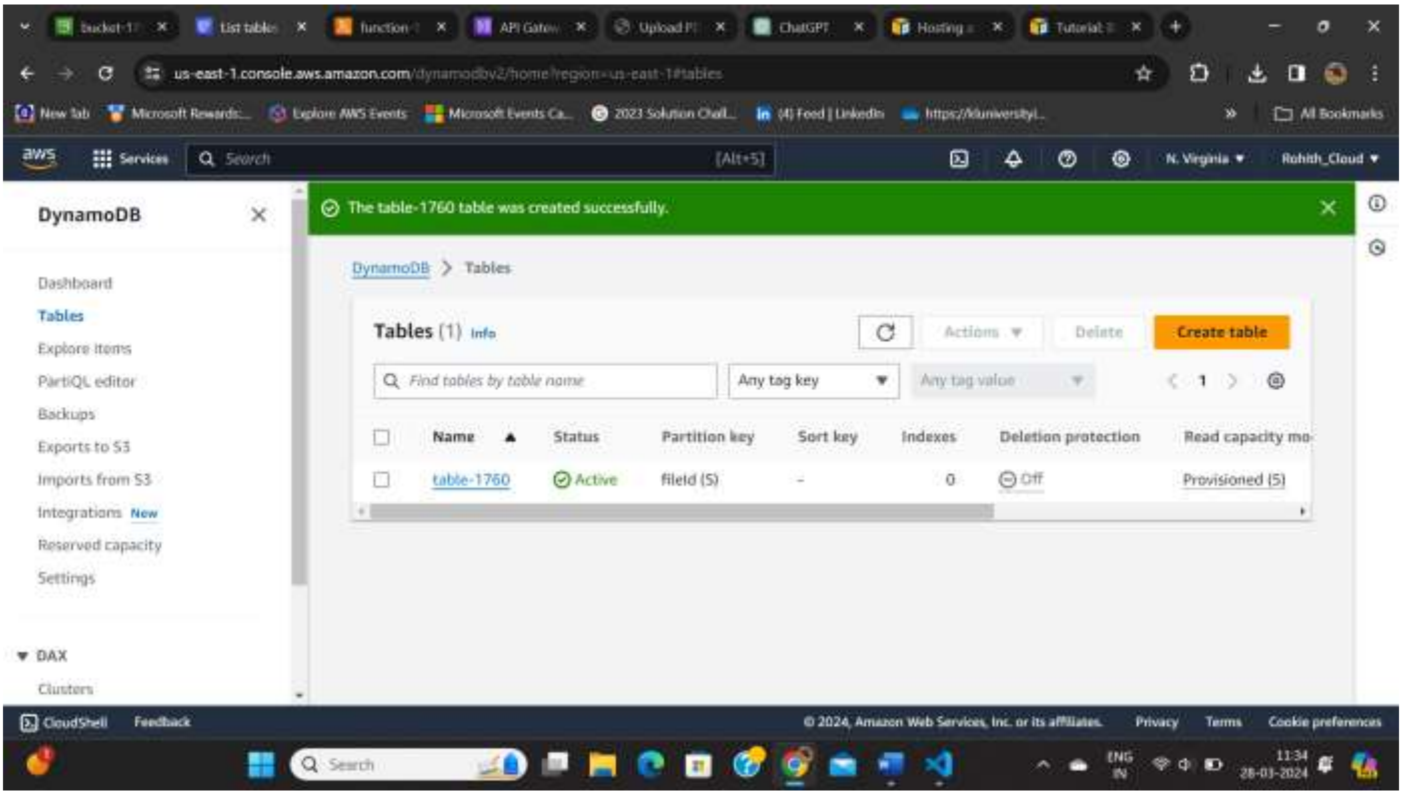


Objects (3) Info

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
index.html	html	March 28, 2024, 10:51:10 (UTC+05:30)	503.0 B	Standard
script.js	js	March 28, 2024, 10:48:11 (UTC+05:30)	1.2 KB	Standard
style.css	css	March 28, 2024, 10:48:12 (UTC+05:30)	1.4 KB	Standard

4. Now, create a DynamoDB table with some name and keep the unique field as fileId.

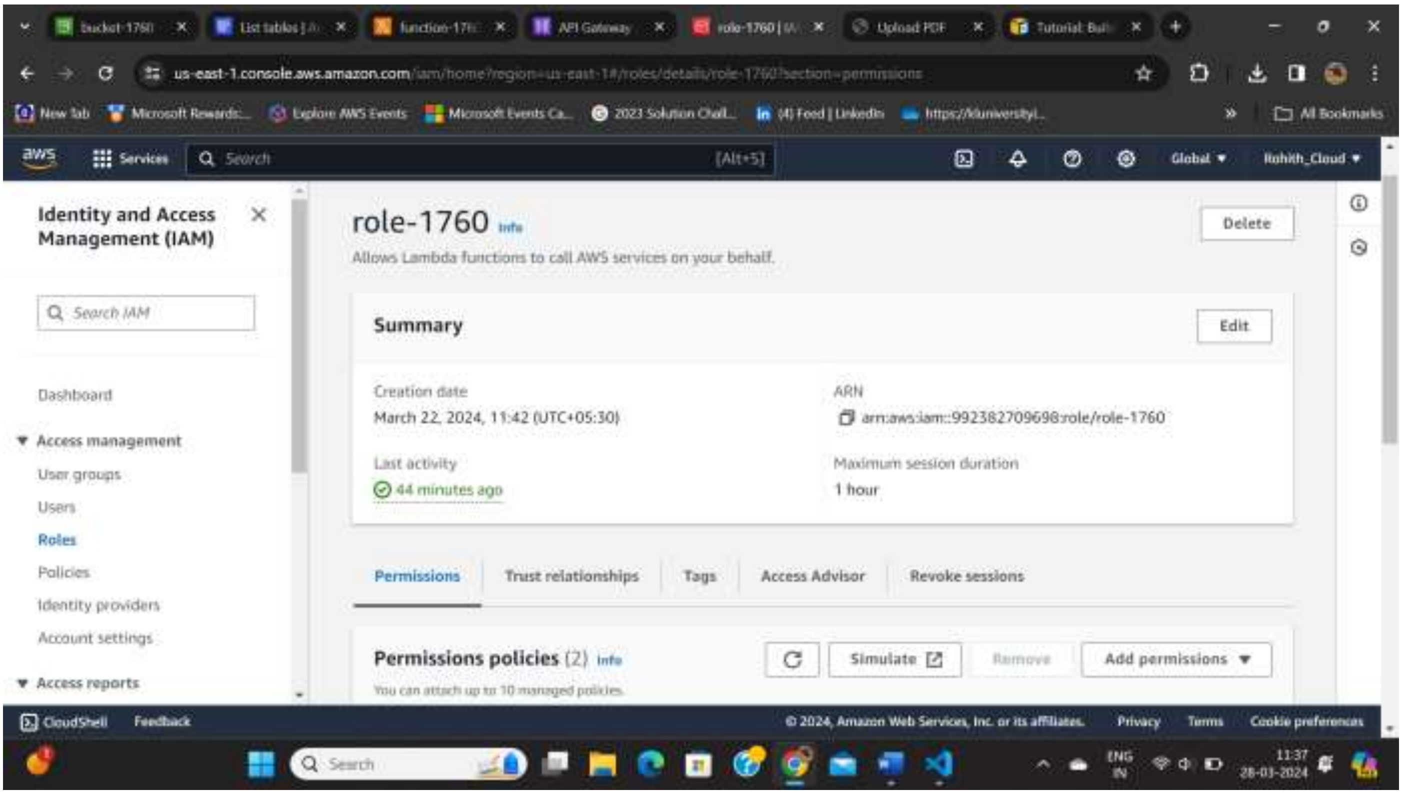


The table-1760 table was created successfully.

Tables (1) Info

Name	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mo
table-1760	Active	fileId (S)	-	0	Off	Provisioned (5)

5. Create a lambda function for our backend with runtime as python.
6. Assign the role of lambda as the role we are going to create now
7. The role will the DynamoDB full access and s3 full access as policies.



us-east-1.console.aws.amazon.com/iam/home?region=us-east-1#/roles/details/role-1760?section=permissions

role-1760 Info

Allows Lambda functions to call AWS services on your behalf.

Summary

Creation date
March 22, 2024, 11:42 (UTC+05:30)

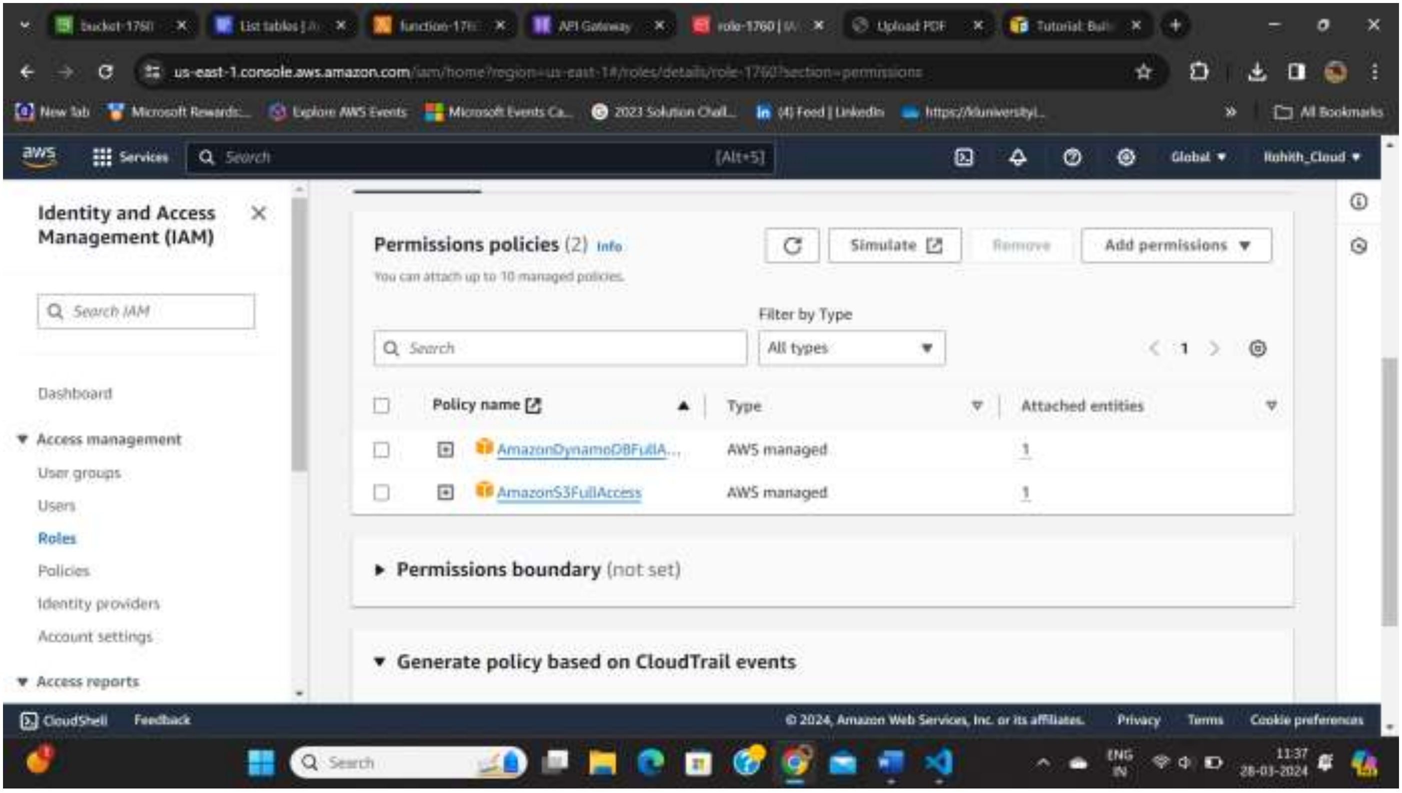
ARN
arn:aws:iam::992382709698:role/role-1760

Last activity
44 minutes ago

Maximum session duration
1 hour

Permissions policies (2) Info

You can attach up to 10 managed policies.



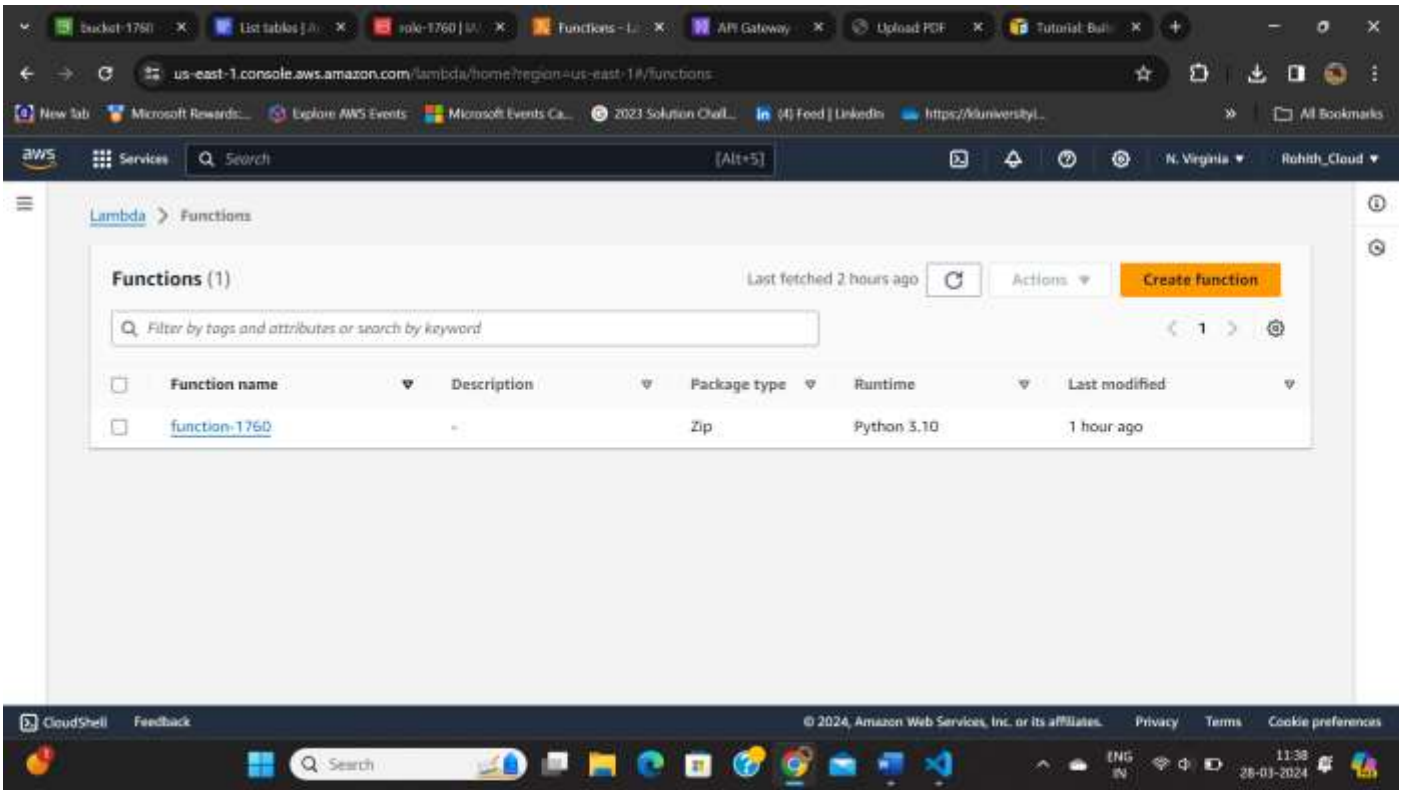
Permissions policies (2) Info

You can attach up to 10 managed policies.

Policy name	Type	Attached entities
AmazonDynamoDBFullA...	AWS managed	1
AmazonS3FullAccess	AWS managed	1

Permissions boundary (not set)

Generate policy based on CloudTrail events



us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions
Services
N. Virginia
Rohith_Cloud
Lambda > Functions
Last fetched 2 hours ago
Actions
Create function
Filter by tags and attributes or search by keyword
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

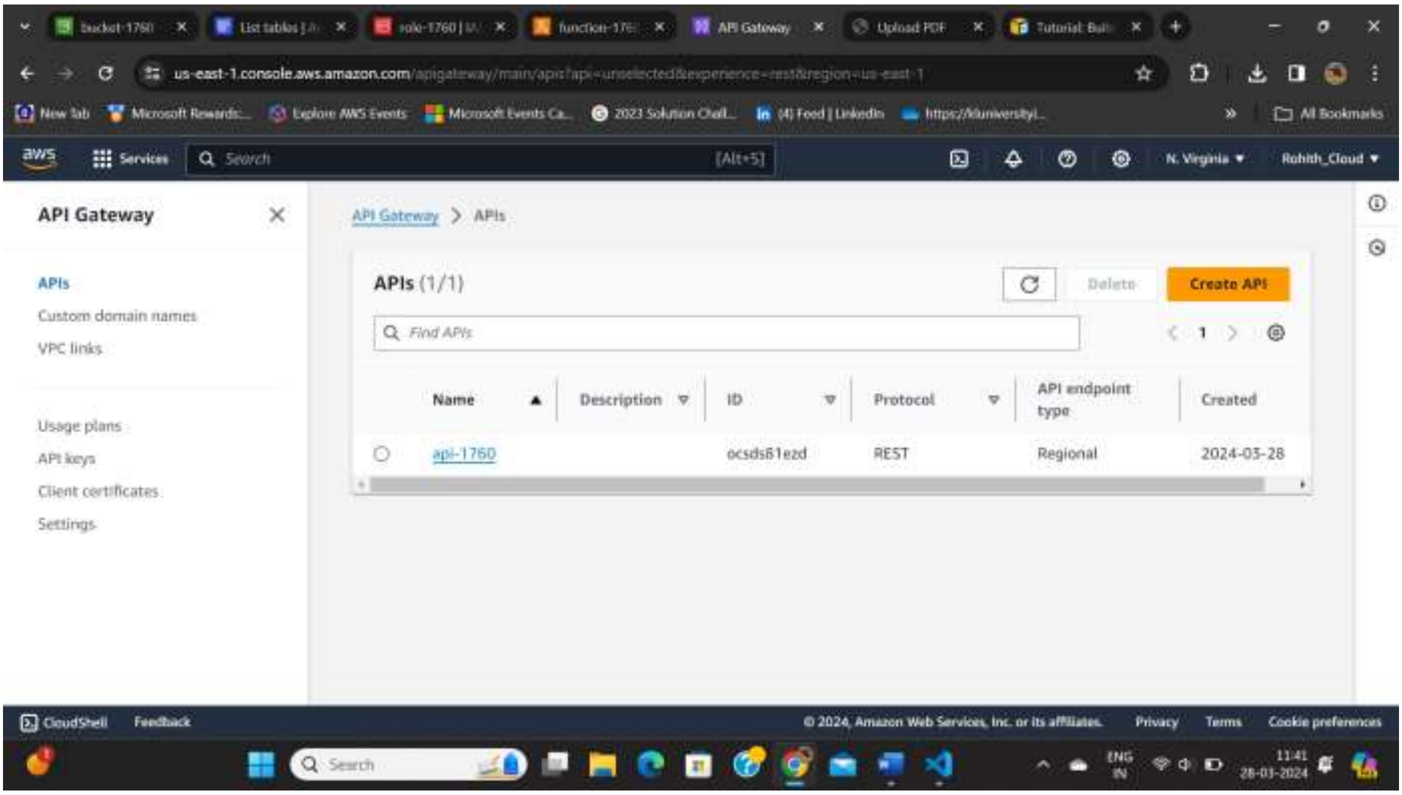
Functions (1)

	Function name	Description	Package type	Runtime	Last modified
□	function-1760	-	Zip	Python 3.10	1 hour ago

Lambda Code:

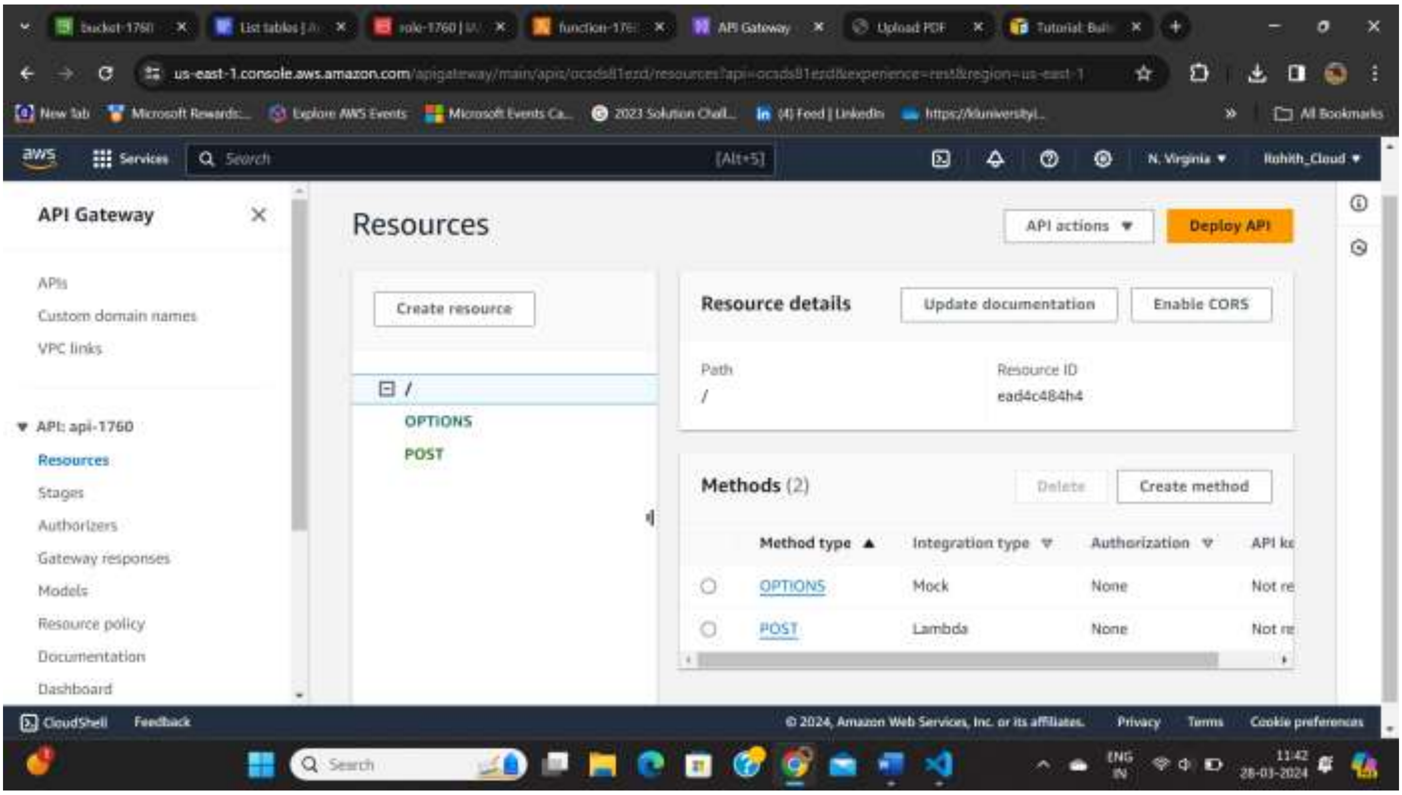
```
import boto3
import base64
from io import BytesIO
from datetime import datetime

s3 = boto3.client('s3')
dynamodb = boto3.client('dynamodb')

def lambda_handler(event, context):
    try:
        operation = event['operation']

        if operation == 'upload':
            file_name = event['fileName']
            file_data = event['fileData'].split(',')[1]

            # Upload file to S3
            s3.upload_fileobj(
                Fileobj=BytesIO(base64.b64decode(file_data)),
                Bucket='bucket-1760',
                Key=file_name
            )
            # Store file details in DynamoDB
            timestamp = datetime.now().isoformat()
            dynamodb.put_item(
                TableName='table-1760',
                Item={
                    'fileId': {'S': file_name},
                    's3Location': {'S': f'https://bucket-1760.s3.us-east-
1.amazonaws.com/{file_name}'},
                    'uploadedAt': {'S': timestamp}
                }
            )

            return {'message': 'File uploaded successfully'}

        elif operation == 'delete':
            file_name = event['fileName']

            # Delete file from S3
            s3.delete_object(
                Bucket='bucket-1760',
                Key=file_name
            )

            # Remove file details from DynamoDB
            dynamodb.delete_item(
                TableName='table-1760',
                Key={
                    'fileId': {'S': file_name}
                }
            )

            return {'message': 'File deleted successfully'}

        elif operation == 'get':
            file_name = event['fileName']

            # Get file from S3
            response = s3.get_object(
                Bucket='bucket-1760',
                Key=file_name
            )
            file_data = response['Body'].read()

            return {'fileData': base64.b64encode(file_data).decode('utf-8')}

        elif operation == 'listFiles':
            # Retrieve list of files from DynamoDB
            response = dynamodb.scan(
                TableName='table-1760'
            )
            files = [item['fileId']['S'] for item in response['Items']]
            return {'files': files}

        else:
            return {'message': 'Invalid operation'}

    except Exception as e:
        print("Error:", e)
        raise Exception('Failed to process operation')
```

8. Create a API Gateway
9. Choose the API type as REST API
10. Build the API
11. Create a method POST for uploading the documents
12. Choose the integration type as lambda and select the lambda function we created earlier.



APIs (1/1)

	Name	Description	ID	Protocol	API endpoint type	Created
○	api-1760		ocsds81ezd	REST	Regional	2024-03-28



Resource details

Path
/

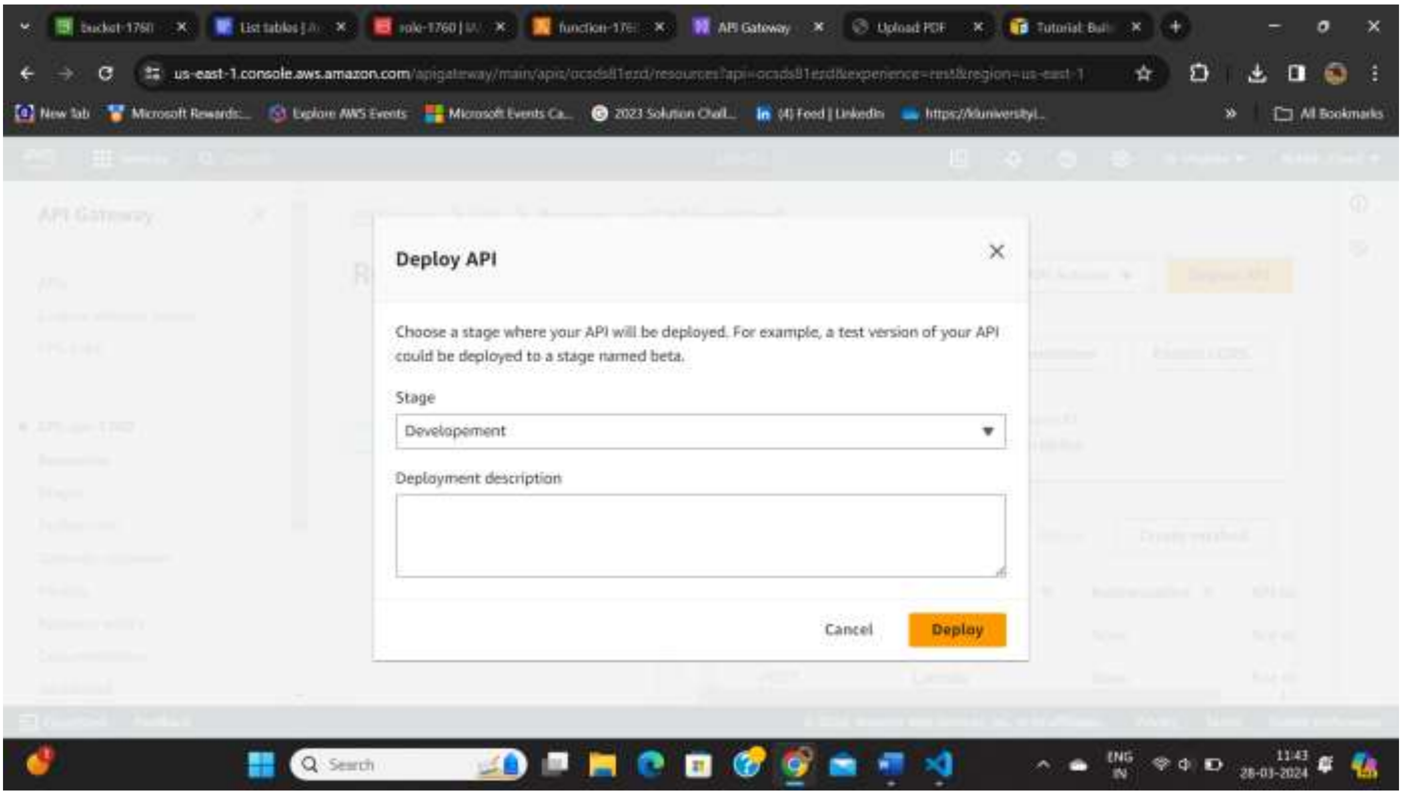
Resource ID
ead4c484h4

Methods (2)

	Method type	Integration type	Authorization	API ke
○	OPTIONS	Mock	None	Not re
○	POST	Lambda	None	Not re

13.Enable CORS for the API

14.Deploy the API by creating a new Stage



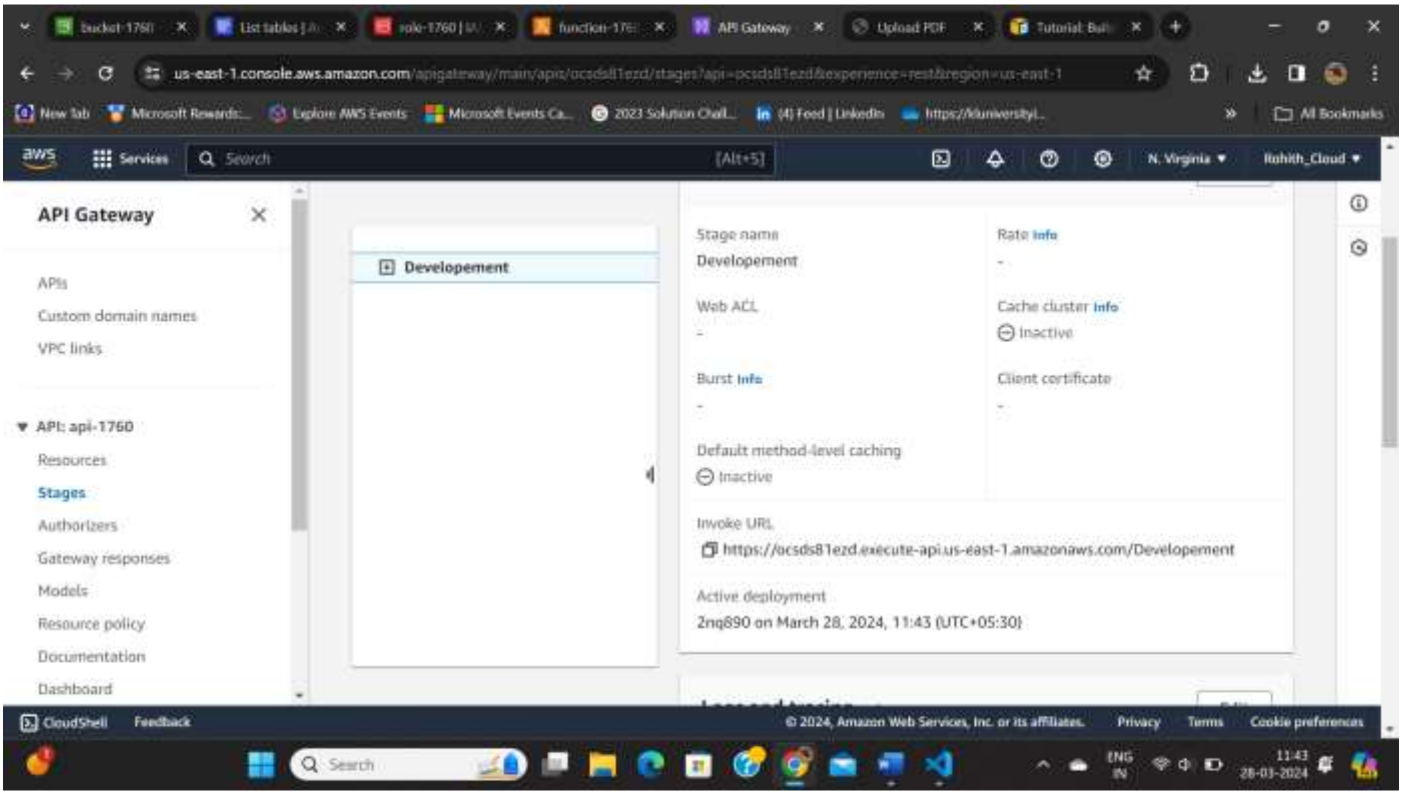
Deploy API

Choose a stage where your API will be deployed. For example, a test version of your API could be deployed to a stage named beta.

Stage
Developement

Deployment description

15.Copy the endpoint of the API and replace it in the lambda function



Stage name
Developement

Rate Info
-

Web ACL
-

Cache cluster Info
Inactive

Burst Info
-

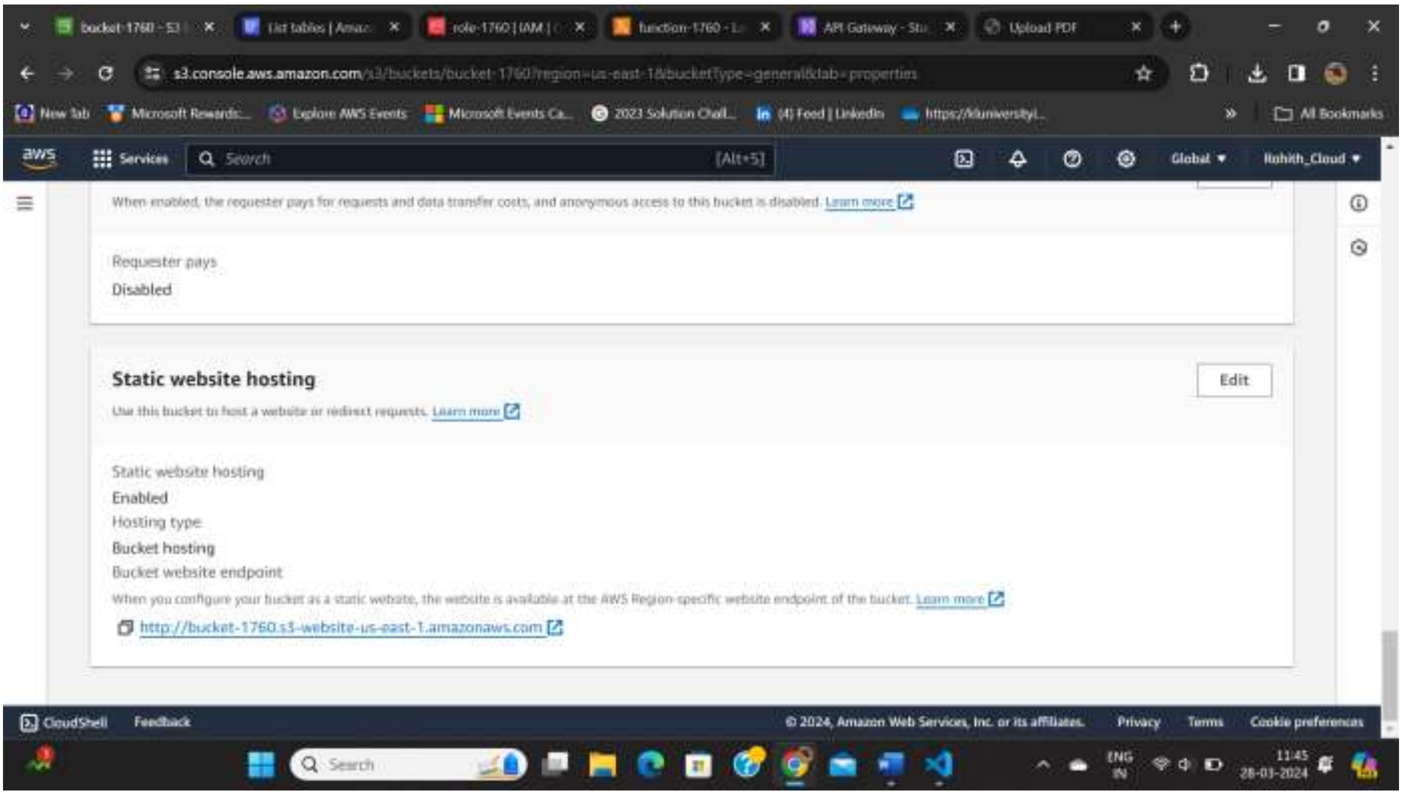
Client certificate
-

Default method-level caching
Inactive

Invoke URL
https://ocsds81ezd.execute-api.us-east-1.amazonaws.com/Developement

Active deployment
2nq890 on March 28, 2024, 11:43 (UTC+05:30)

16.Click the endpoint of the S3 bucket to open your website



s3.console.aws.amazon.com/s3/buckets/bucket-1760?region=us-east-1&bucketType=general&tab=properties
Edit

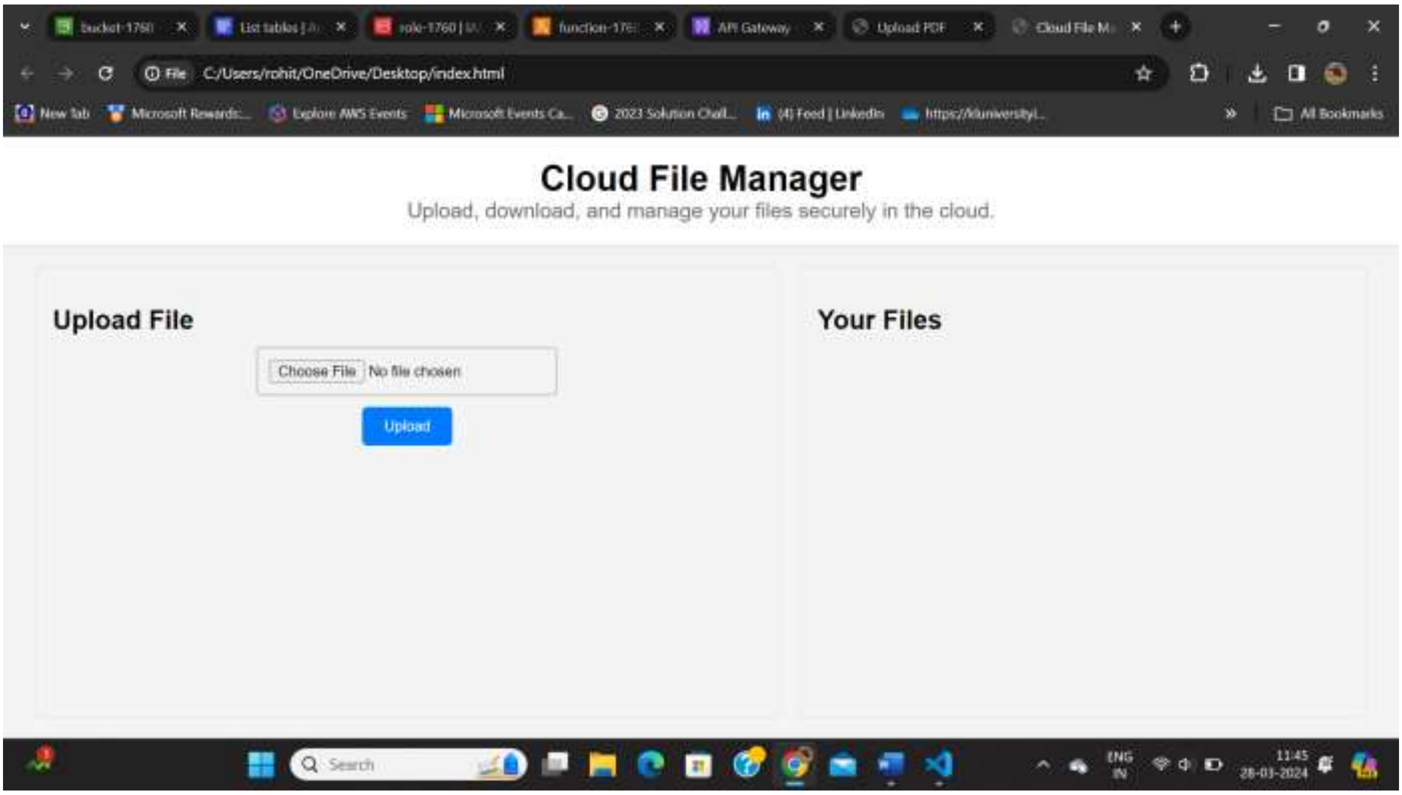
When enabled, the requester pays for requests and data transfer costs, and anonymous access to this bucket is disabled. Learn more

Requester pays
Disabled

Static website hosting

Use this bucket to host a website or redirect requests. Learn more

Static website hosting
Enabled
Hosting type
Bucket hosting
Bucket website endpoint
When you configure your bucket as a static website, the website is available at the AWS Region-specific website endpoint of the bucket. Learn more
http://bucket-1760.s3-website-us-east-1.amazonaws.com



C:/Users/rohit/OneDrive/Desktop/index.html

Cloud File Manager

Upload, download, and manage your files securely in the cloud.

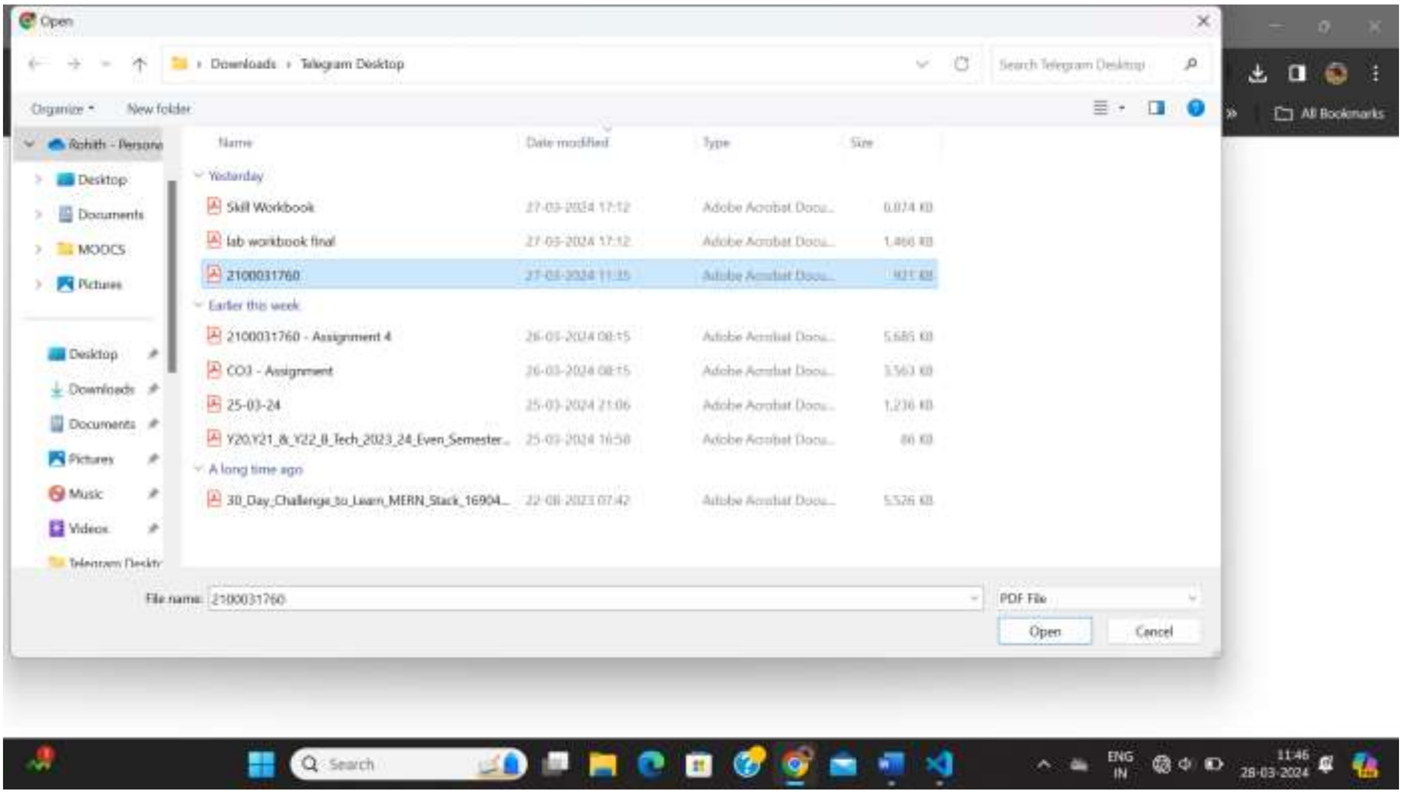
Upload File

Choose File No file chosen

Upload

Your Files

17.Click upload to upload your files



Downloads › Telegram Desktop

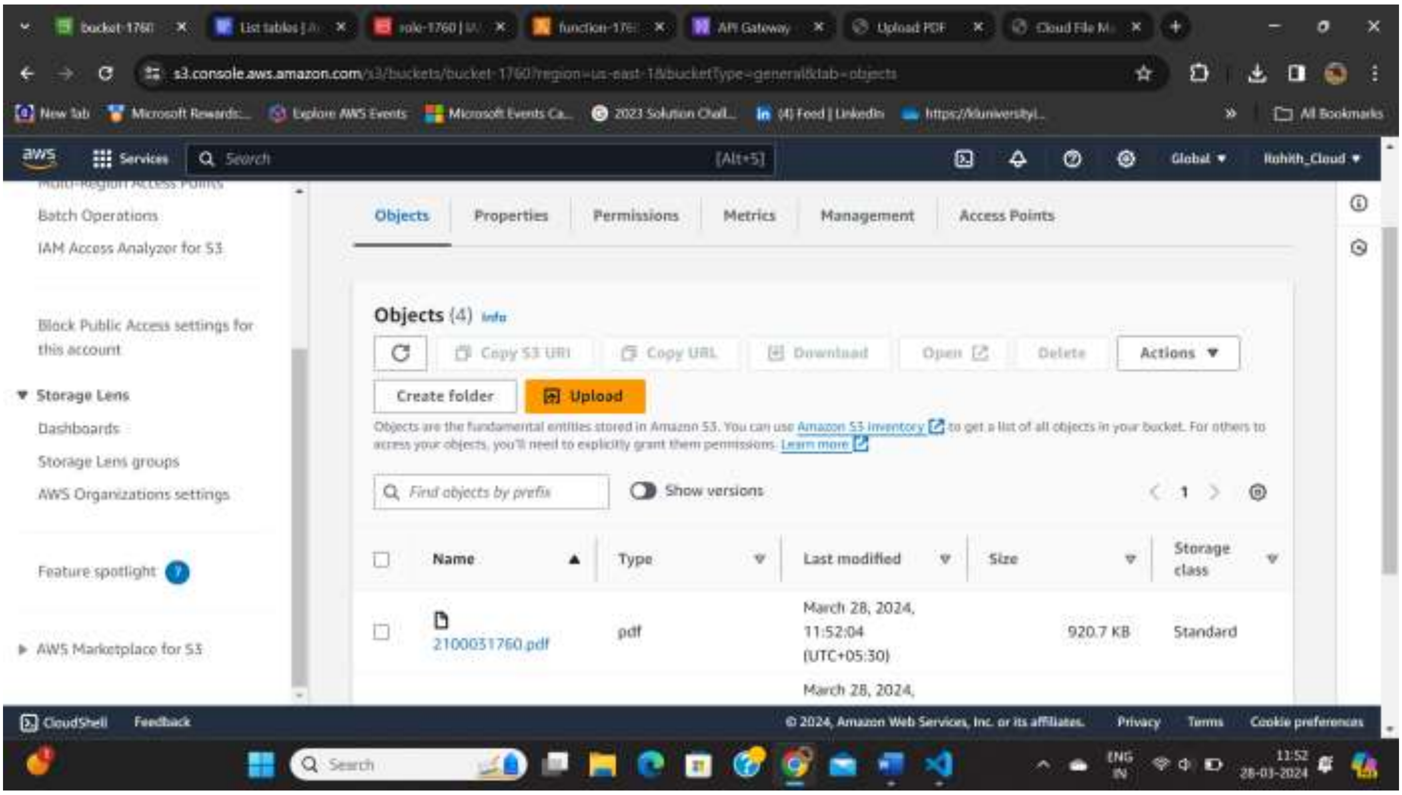
Name	Date modified	Type	Size
Yesterday			
Skill Workbook	27-03-2024 17:12	Adobe Acrobat Docu...	6,074 KB
lab workbook final	27-03-2024 17:12	Adobe Acrobat Docu...	1,466 KB
2100031760	27-03-2024 11:35	Adobe Acrobat Docu...	921 KB
Earlier this week			
2100031760 - Assignment 4	26-03-2024 08:15	Adobe Acrobat Docu...	5,685 KB
CO3 - Assignment	26-03-2024 08:15	Adobe Acrobat Docu...	3,563 KB
25-03-24	25-03-2024 21:06	Adobe Acrobat Docu...	1,236 KB
Y20,Y21_&_Y22_B_Tech_2023_24_Even_Semester...	25-03-2024 16:58	Adobe Acrobat Docu...	86 KB
A long time ago			
30_Day_Challenge_to_Learn_MERN_Stack_16904...	22-08-2023 07:42	Adobe Acrobat Docu...	5,526 KB

File name: 2100031760

PDF File

Open Cancel

18.Your file is uploaded into the bucket

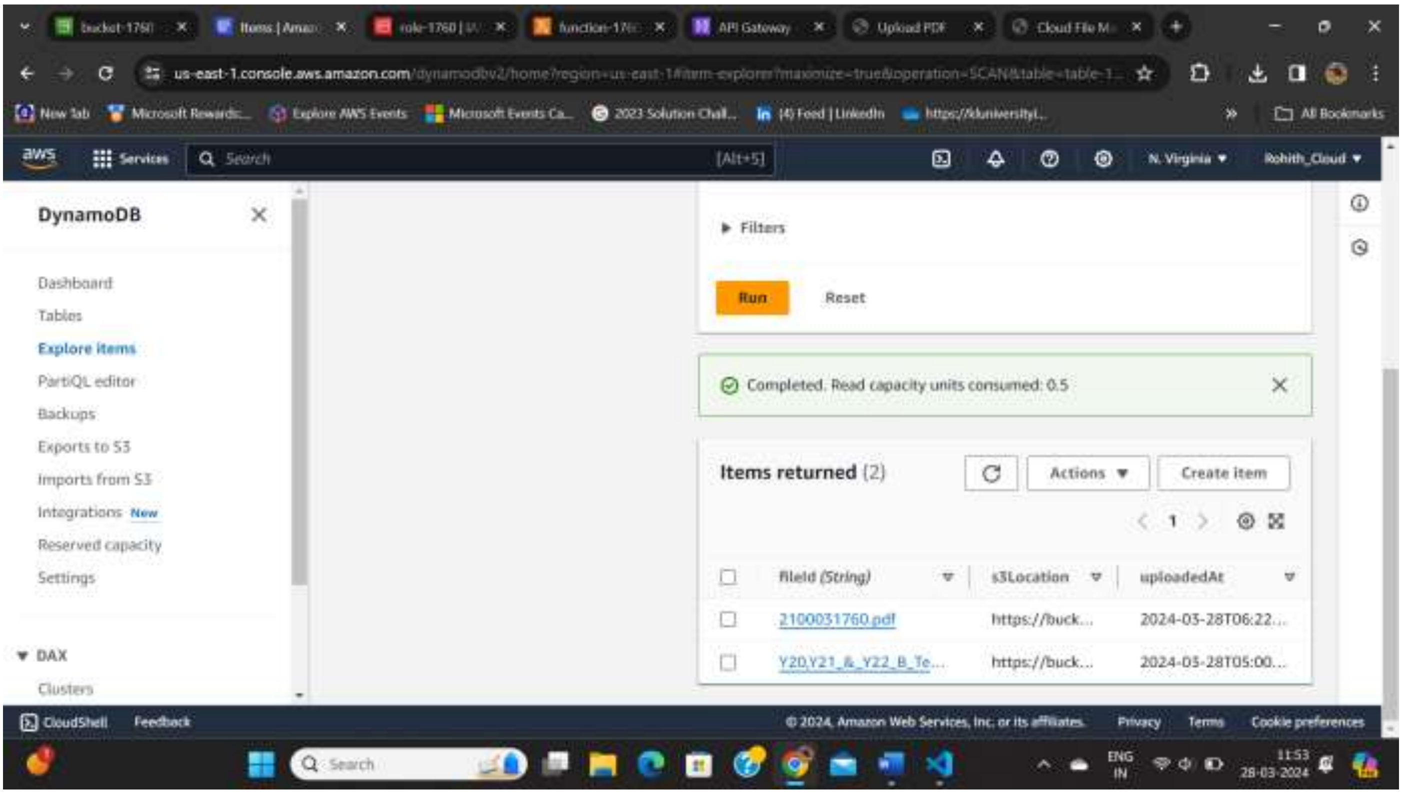


Objects (4) Info

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
2100031760.pdf	pdf	March 28, 2024, 11:52:04 (UTC+05:30)	920.7 KB	Standard
		March 28, 2024,		

19.Now check the DynamoDB table for you file location.

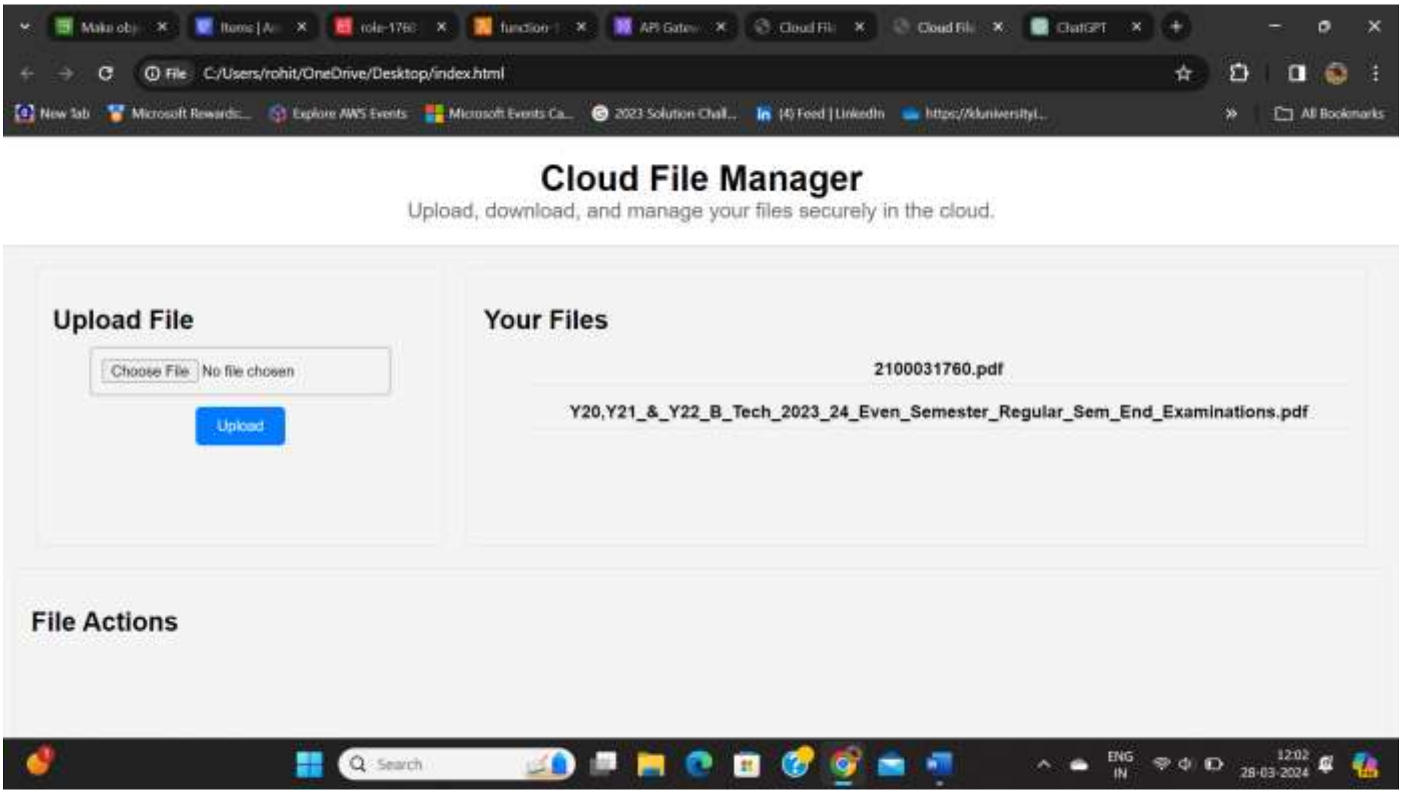


Completed. Read capacity units consumed: 0.5

Items returned (2)

fileId (String)	s3Location	uploadedAt
2100031760.pdf	https://buck...	2024-03-28T06:22...
Y20,Y21_&_Y22_B_Te...	https://buck...	2024-03-28T05:00...

20.We use this address to perform operations on our file



Cloud File Manager

Upload, download, and manage your files securely in the cloud.

Upload File

Your Files

2100031760.pdf

Y20,Y21_&_Y22_B_Tech_2023_24_Even_Semester_Regular_Sem_End_Examinations.pdf

File Actions